

1 Einleitung

1.1 Was ist Informatik?

1.2 Leistungssteigerung von Rechnern

1.3 Turing Test und Eliza

1.4 Magische Quadrate

1.5 Euklids Algorithmus

1.6 Zusammenfassung

Was ist Informatik?

Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen. Sie befaßt sich mit der Struktur, den Eigenschaften und den Beschreibungsmöglichkeiten von Information und informationsverarbeitenden Systemen.

Die vier Schwerpunkte der Informatik:

Theorie
Rechenanlagen Programmierung
Anwendungen

Teilgebiete der Informatik:

**Theoretische Informatik,
Praktische Informatik,
Angewandte Informatik,
Technische Informatik.**

Das Wort "Informatik":

Das Wort Informatik ist ein Kunstwort, zusammengesetzt aus den Wörtern Information und Automatik. In Deutschland wurde es wahrscheinlich erstmals von Karl Steinbuch im Jahre 1957 benutzt. In dem Aufsatz "INFORMATIK: Automatische Informationsverarbeitung, SEG-Nachrichten 1957, Heft 4, S. 171-176" beschreibt Steinbuch den Umfang eines Gebietes Informatik. 1968 umreißt Forschungsminister Gerhard Stoltenberg in einer Berliner Rede ein Förderprogramm für eine neue Wissenschaft, der er den Namen Informatik gibt. Hierbei stützt er sich auf das französische Wort "Informatique", das 1962 von Philippe Dreyfus zur Bezeichnung der Unternehmung "Société d'Informatique Appliquée" genutzt wurde. 1967 erfolgt eine Definition des Begriff Informatique durch die Académie Française.

Auswahl aus den Bindestrich-Informatiken:

Bio-Informatik,
Geo-Informatik,
Medien-Informatik,
Medizin-Informatik,
Quanteninformatik,
Rechts-Informatik,
Umwelt-Informatik,
Wirtschafts-Informatik.

In ihrem Buch "Parallel Computers 2" geben Hockney und Jesshope einen Faktor von 10 in 5 Jahren für die Steigerung der Rechengeschwindigkeit von Computern an. Diesen Wert leiten sie ab aus der Veränderung der Ausführungsgeschwindigkeit der Multiplikationsoperation im Zeitraum 1950 bis 1975.

| Jahr | Steigerung |
|------|-------------|
| 0 | 1,00 |
| 1 | 1,58 |
| 2 | 2,51 |
| 3 | 3,98 |
| 4 | 6,31 |
| 5 | 10 |
| 6 | 15,85 |
| 7 | 25,12 |
| 8 | 39,81 |
| 9 | 63,10 |
| 10 | 100 |
| 11 | 158,49 |
| 12 | 251,19 |
| 13 | 398,11 |
| 14 | 630,96 |
| 15 | 1.000 |
| 20 | 10.000 |
| 30 | 1.000.000 |
| 40 | 100.000.000 |

Für die Entwicklung der Taktzyklen geben Hockney und Jesshope folgende Zahlen:

| Rechner | Jahr | Zykluszeit |
|----------|------|------------|
| EDSAC-1 | 1949 | 2000 ns |
| ACE | 1951 | 1000 ns |
| CDC-6600 | 1964 | 100 ns |
| CDC-7600 | 1969 | 27,5 ns |
| CRAY-1 | 1976 | 12,5 ns |
| CRAY-2 | 1984 | 4 ns |

“Moore’ Law”:

$$(\text{circuits per chip}) = C_{1975} * 2^{(\text{year} - 1975) / 1,5}$$

Allgemeine Interpretation:

Die zu einem festen Preis kaufbare Rechenleistung verdoppelt sich alle 18 Monate.

Ursprüngliche Formulierung von Gordon E. Moore im Artikel: "Cramming More Components Onto Integrated Circuits" im Magazin Electronics (Vol. 38, No. 8, April 19, 1965):

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will remain nearly constant for at least 10 years."

Tabelle zur erwarteten Steigerung der Rechenleistung:

| | |
|---------|------|
| Jahr 1: | 1,6 |
| Jahr 2: | 2,2 |
| Jahr 3: | 4 |
| Jahr 4: | 6,5 |
| Jahr 5: | 10,1 |
| Jahr 6: | 16 |
| Jahr 7: | 25,4 |
| Jahr 8: | 40,3 |
| Jahr 9: | 64 |

Illustration zu Moores Beobachtung:

Beispiel: Entwicklung der Intelprozessoren
(Daten von Intels Homepage)

| Jahr | Rechner | Transistorenzahl |
|------|-------------|------------------|
| 1971 | 4004 | 2.300 |
| 1972 | 8008 | 2.500 |
| 1974 | 8080 | 4.500 |
| 1978 | 8086 | 29.000 |
| 1982 | 286 | 134.000 |
| 1985 | i386 | 275.000 |
| 1989 | i486 | 1.200.000 |
| 1993 | Pentium | 3.100.000 |
| 1997 | Pentium II | 7.500.000 |
| 1999 | Pentium III | 9.500.000 |
| 2000 | Pentium 4 | 42.000.000 |
| 2001 | Itanium | 25.000.000 |
| 2002 | Itanium 2 | 220.000.000 |
| 2004 | Itanium 2 | 592.000.000 |

Eine Lösung der Gleichung $x^{33} = 592000000 / 2300$ ergibt $x = 1,45867$.

Kennzahlen eines Hochleistungsrechners:

Name: NEC Earth Simulator

Zahl der Prozessoren: 5120
aufgeteilt auf 640 Knoten zu je 8 Prozessoren,
Zykluszeit eines Prozessors 2 ns,
Leistung eines Prozessors 8 GFlop/s.

Hauptspeicher: 10 TB,
pro Knoten 16 GB gemeinsamer Speicher.

Plattenspeicher: 700 TB

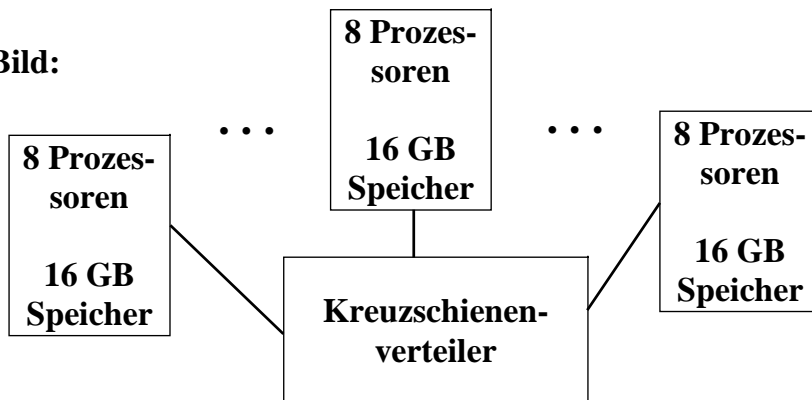
Massenspeicher: 1,6 PB

Theoretische Gesamtleistung: 40 TFlop/s

Erreichbare Gesamtleistung: 36 TFlop/s

Verbindungsnetz: Kreuzschienenverteiler

Bild:



Geschwindigkeitssteigerung beim Problemlösen:

Tabelle aus Raj Reddy und Allen Newell:

Multiplicative Speedup of Systems
in *Perspectives on Computer Science* (A. K. Jones),
Academic Press, 1977.

Beobachtete Steigerungsfaktoren der Ausführungsgeschwindigkeit von Programmen auf Grund von Spartenfortschritten:

| | |
|----------------------|------------|
| Technologie | 20 bis 50 |
| Architektur | 2 bis 10 |
| Systemsoftware | 2 bis 10 |
| Programmorganisation | 2 bis 20 |
| Algorithmen | 2 bis 20 |
| Reimplementation | 2 bis 10 |
| Problemwissen | 10 bis 100 |
| Heuristiken | 10 bis 100 |

Bemerkung: Falls die einzelnen Steigerungen unabhängig voneinander sind, sind Geschwindigkeitssteigerungen um bis zu elf Zehnerpotenzen möglich.

Präfixe für Einheiten im Dezimalsystem:

| Faktor | Name | Symbol |
|------------|-------|--------|
| 10^{24} | yotta | Y |
| 10^{21} | zetta | Z |
| 10^{18} | exa | E |
| 10^{15} | peta | P |
| 10^{12} | tera | T |
| 10^9 | giga | G |
| 10^6 | mega | M |
| 10^3 | kilo | k |
| 10^2 | hecto | h |
| 10^1 | deka | da |
| 10^{-1} | deci | d |
| 10^{-2} | centi | c |
| 10^{-3} | milli | m |
| 10^{-6} | micro | μ |
| 10^{-9} | nano | n |
| 10^{-12} | pico | p |
| 10^{-15} | femto | f |
| 10^{-18} | atto | a |
| 10^{-21} | zepto | z |
| 10^{-24} | yocto | y |

Präfixe im Dualsystem:

Standard: IEC 60027-2, Second edition, 2000-11, Letter symbols to be used in electrical technology – Part 2: Telecommunications and electronics.

| Faktor | Kurzname | Symbol | Langname |
|----------|----------|--------|------------|
| 2^{10} | kibi | Ki | kilobinary |
| 2^{20} | mebi | Mi | megabinary |
| 2^{30} | gibi | Gi | gigabinary |
| 2^{40} | tebi | Ti | terabinary |
| 2^{50} | pebi | Pi | petabinary |
| 2^{60} | exbi | Ei | exabinary |

Beispiele:

| | | |
|-----------|---|-------------------|
| 1 kibibit | = | 1.024 bit |
| 1 kilobit | = | 1.000 bit |
| 1 mebibit | = | 1.048.576 bit |
| 1 megabit | = | 1.000.000 bit |
| 1 gibibit | = | 1.073.741.824 bit |
| 1 gigabit | = | 1.000.000.000 bit |

Turing Test:

Können Computer denken?

Im Jahre 1950 schlug Alan Turing zur Beantwortung der obigen Frage folgende Testsituation vor. Ein Mensch führt ein Gespräch mit einem Gegenüber; kann er nicht aufgrund des Gespräches entscheiden, ob sein Gesprächspartner ein Mensch oder eine Maschine ist, dann ist dem Gegenüber menschliche Intelligenz zuzusprechen.

Turing vermutete, daß ein Rechner mit einer Speicherkapazität von 1 Gigabit, dessen Existenz er für das Jahr 2000 erwartete, einen Durchschnittsmenschen in 70 % der Fälle täuschen würde.

Als Turing Test läßt sich das Programm Eliza ansehen, das Joseph Weizenbaum um das Jahr 1963 schuf. Später schrieb er dazu:

"I was startled to see how very deeply people conversing with the program became emotionally involved with the computer and how unequivocally they anthropomorphized it. Once my secretary, who had watched me work on the program for many months and therefor surely knew it to be merely a computer program, started conversing with it. After only a few exchanges, she asked me to leave the room."

Eine Anekdote von Daniel Bobrow:

One Saturday morning about 9 a.m., a Vice President of Bolt, Beranek, and Newman in charge of selling our Telcomp commercial service arrived at our PDP-1 computer room hoping to be able to use the computer for a demonstration. He found a note on the PDP-1 teletype that read: "TYPE ALL MESSAGES INTO THIS TELETYPE AND END THEM WITH A PERIOD." This note was left there by one of the programming bums who had just succeeded in making run a LISP version of the DOCTOR program (based on ELIZA as developed by Weizenbaum). However, the VP thought that this note indicated that this teletype was linked with the one in my home, since I often use the computer all night. Therefore, he started typing to "me," but in fact was typing to this version of the DOCTOR program. The following conversation ensued:

VP: Do you think that I can use the computer this morning?

Eliza: Why do you ask?

VP: I might be able to make some additional Telcom sales.

Eliza: Why aren't you so sure?

VP: My perspective customers have never seen a demonstration of the system.

Eliza: Does this mean very much to you?

VP: Of course it does.

Eliza: You seem very positive about that.

VP: This is ridiculous.

Eliza: What do you mean, this is ridiculous?

VP: Is there any reason why I can't use the computer now?

Eliza: Tell me more about your perspective customers.

VP: Please dial me up on 491-1850

Note that after the last remark the VP did not obey instructions, and left out the period. Therefore, of course, the computer didn't answer him. This so infuriated the VP, who thought I was playing games with him, that he called me up, woke me from a deep sleep, and said:

VP: Why are you being so snotty to me?

Bobrow: What do you mean I am being snotty to you?

The VP angrily read me the dialog that "we" had been having, and couldn't get any response but laughter from me. It took a while to convince him it really was a computer.

Ein Beispieldialog mit Eliza:

H: *Men are all alike.*

Eliza: In what way?

H: *They're always bugging us about something or other.*

Eliza: Can you think of a specific example?

H: *Well, my boyfriend made me come here.*

Eliza: Your boyfriend made you come here?

H: *He says I'm depressed much of the time.*

Eliza: I am sorry to hear that you are depressed.

H: *It's true. I am unhappy.*

Eliza: Do you think coming here will help you not to be unhappy?

H: *I need some help, that much seems certain.*

Eliza: What would it mean to you if you got some help?

H: *Perhaps I could learn to get along with my mother.*

Eliza: Tell me more about your family.

H: *My mother takes care of me.*

Eliza: Who else in your family takes care of you?

H: *My father.*

Eliza: Your father.

H: *You are like my father in some ways.*

Eliza: What resemblance do you see?

H: *You are not very aggressive but I think you don't want me to notice that.*

Eliza: What makes you think I am not very aggressive?

H: *You don't argue with me.*

Eliza: Why do you think I don't argue with you?

H: *You are afraid of me.*

Eliza: Does it please you to believe I am afraid of you?

H: *My father is afraid of everybody.*

Eliza: What else comes to your mind when you think of your father?

H: *Bullies.*

Eliza: Does that have anything to with the fact that your boyfriend made come you here?

Zum Aufbau von Eliza:

Eliza besteht aus zwei Teilen, einem allgemeinen Verwaltungsteil und einem austauschbaren Rollenskript. Das Skript enthält zu einer Rolle passende Schlüsselwörter. Für die Rolle des Psychotherapeuten sind dies Wörter wie Mutter, Vater, Traum, Angst, u. s. w. Das Programm Eliza sucht nun in der Eingabe nach dem Auftreten dieser Wörter; es führt eine Tabelle mit vorgestanzten Antwortmustern und nutzt diese zur Bildung intelligent klingender Antworten.

Beispiel:

Muster: ?x my mother ?y

Antworten: Who else in your family ?y?
 Tell me more about your family.

Möglicher Eingabesatz: Yesterday I was miserable when
 my mother told me I was a disappointment.

Antwortsatz nach Anredeanpassung: Who else in your
family told you you were a disappointment?

Findet Eliza kein Schlüsselwort in der Eingabe, dann
benutzt es gesprächserhaltende Aussagen wie:

What makes you say that?
Could you be more specific?

...

Magische Quadrate:

Der Kupferstich Melancolia von Albrecht Dürer zeigt das folgende 4 x 4 – Zahlentableau:

| | | | |
|----|----|----|----|
| 16 | 3 | 2 | 13 |
| 5 | 10 | 11 | 8 |
| 9 | 6 | 7 | 12 |
| 4 | 15 | 14 | 1 |

Einige Beobachtungen:

Teilt man das 4 x 4 – Quadrat in vier 2 x 2 – Quadrate,
dann ist die Summe der Zahlen in jedem kleineren
Quadrat 34.

Die Zahlensumme für das Mittelquadrat ergibt 34.

Die Summe der Zahlen in den vier Ecken ist auch 34.

Magische Quadrate:

Aufgabe: Die Zahlen $1, 2, \dots, n^2$ sind so in einem quadratischen Tableau der Seitenlänge n anzuordnen, daß gilt:

- Summe über jede Spalte
- = Summe über jede Zeile
- = Summe über Hauptdiagonale
- = Summe über Nebendiagonale

Beispiel: $n = 3$

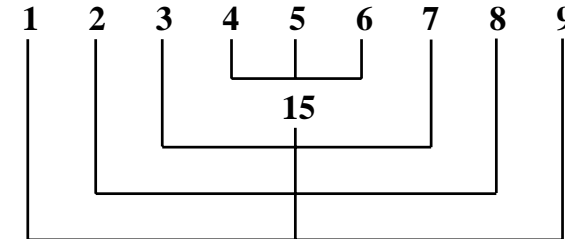
Zahlen: 1, 2, 3, 4, 5, 6, 7, 8, 9

Symmetrievermutung:

Mittelzahl ins Mittelfeld

| | | |
|--|---|--|
| | | |
| | 5 | |
| | | |

Die magische Zahl ist $15 = 45 / 3$. Es lassen sich vier einfache Tripel der Summenzahl 15 bilden, an denen 5 beteiligt ist.



Eintrag der obiger Zahlentripel in das Tableau ergibt:

| | | |
|---|---|---|
| 4 | 3 | 8 |
| 9 | 5 | 1 |
| 2 | 7 | 6 |

Bemerkung: Ein Überprüfen aller $9! = 362.880$ Anordnungen der Zahlen 1 bis 9 im 3×3 -Tableau zeigt, bis auf Symmetrien ist obiges Quadrat die einzige Lösung.

Anleitung zur Erzeugung magischer Quadrate ungerader Ordnung:

1. Zeichne Quadrat der gewünschten Größe.
2. Setze Zahl auf 1.
3. Schreibe Zahl in die Mitte der ersten Zeile.
4. Erhöhe Zahl um 1.
5. Gehe einen Schritt nach rechts und einen Schritt nach oben; es können nun folgende Fälle eintreten:
 - a) Platz ist frei und gehört zum Quadrat;
 - b) Platz ist belegt und gehört zum Quadrat;
 - c) Quadrat wird nach rechts verlassen;
 - d) Quadrat wird nach oben verlassen.
6. Führe in den Fällen b), c) und d) folgende Korrekturen durch:
 - b) Gehe zurück und eine Zeile tiefer.
 - c) Gehe in gleicher Zeile zur ersten Spalte.
 - d) Gehe in gleicher Spalte zur letzten Zeile.
7. Trage Zahl in gefundene Position ein.
8. Falls das Quadrat noch nicht gefüllt ist, dann gehe zu Anweisung 4, sonst beende Algorithmus.

Bemerkung: Die obige Anleitung enthält mindestens eine Ungenauigkeit.

Das gleiche Verfahren kann man benutzen, um magische Multiplikationsquadrate zu erzeugen.

Ein Beispiel:

| | | | | |
|-----|-----|------|-----|-----|
| 54 | 648 | 1 | 12 | 144 |
| 324 | 16 | 6 | 72 | 27 |
| 8 | 3 | 36 | 432 | 162 |
| 48 | 18 | 216 | 81 | 4 |
| 9 | 108 | 1296 | 2 | 24 |

$$\text{magische Zahl} = 2^{10} * 3^{10} = 60.466.176$$

Bemerkung: In gewisser Weise ist die Bildung dieses Quadrates einfacher als die Bildung eines magischen Additionsquadrates, denn die Exponentenpaare (x, y) müssen nur so angeordnet werden, daß in jeder Zeile und jeder Spalte jeder Exponent genau einmal auftritt. Die Wahl des Startpunktes sorgt für die korrekte Berücksichtigung der beiden Diagonalen.

```
// Erzeugung magischer Quadrate ungerader Ordnung
```

```
#include <iostream>      // Ein- und Ausgabe
#include <iomanip>       // Formatierung
using namespace std;

int** matrix;          // globale Matrix

int generate (int);    // Deklaration der
                     // Erzeugungsfunktion

int main () {         // Hauptprogramm

    // Größe des Quadrates
    int groesse = 9;
    // Einlesen der Größe
    cerr << "Größe = ";
    cin >> groesse;

    cout << "Magisches Quadrat der Größe "
         << groesse
         << ':'
         << endl << endl;

    if (generate (groesse) == 1) {
        cout << "Fehler: Quadratgröße nicht"
             << " positiv und ungerade"
             << endl << endl;
        return 0;
    } else {
```

```
        for (int i = 0; i < groesse; ++i) {
            for (int j = 0; j < groesse; ++j)
                cout << setw (4)
                    << matrix [i] [j];
            cout << endl;
        }
    }
```

```
    // Berechnung der magischen Zahl
    int magzahl = 0;
```

```
    for (int i = 0; i < groesse; ++i)
        magzahl += matrix [0] [i];
```

```
    cout << endl
         << "Magische Zahl = "
         << magzahl
         << endl << endl;
```

```
    return 0;
} //main
```

```
int generate (int g) {
    // Keine Größenprüfung; nur Prüfung, ob Parameter
    // prinzipiell gültig
    if (g < 0 || g % 2 == 0)
        return 1; // Fehleranzeige
```

```
    // Speicherzuweisung für Matrix
    matrix = new int* [g];
    for (int i = 0; i < g; ++i)
        matrix [i] = new int [g];
```

```

// Initialisierung der Matrix
for (i = 0; i < g; ++i) // Mangel des MS-Compilers
for (int j = 0; j < g; ++j)
    matrix [i] [j] = 0;

int zahl = 1;
int zeile = 0;
int spalte = g/2;

// Füllen des Quadrates
for (int j = 0; j < g*g; ++j) {
    matrix [zeile] [spalte] = zahl;
    // Berechnung neuer Zeile und Spalte
    int zeile0 = (zeile + g - 1) % g;
    int spalte0 = (spalte + 1) % g;
    ++zahl;
    // Prüfung, ob Feld schon besetzt.
    if (matrix [zeile0] [spalte0] == 0) {
        spalte = spalte0;
        zeile = zeile0;
    } else {
        // Vertrauen auf Einfachheit des
        // Algorithmus
        zeile = zeile + 1;
    }
}
return 0; //erfolgreich
} //generate

```

Beispielausgaben:

Magisches Quadrat der Größe 5:

| | | | | |
|----|----|----|----|----|
| 17 | 24 | 1 | 8 | 15 |
| 23 | 5 | 7 | 14 | 16 |
| 4 | 6 | 13 | 20 | 22 |
| 10 | 12 | 19 | 21 | 3 |
| 11 | 18 | 25 | 2 | 9 |

Magische Zahl = 65

Magisches Quadrat der Größe 9:

| | | | | | | | | |
|----|----|----|----|----|----|----|----|----|
| 47 | 58 | 69 | 80 | 1 | 12 | 23 | 34 | 45 |
| 57 | 68 | 79 | 9 | 11 | 22 | 33 | 44 | 46 |
| 67 | 78 | 8 | 10 | 21 | 32 | 43 | 54 | 56 |
| 77 | 7 | 18 | 20 | 31 | 42 | 53 | 55 | 66 |
| 6 | 17 | 19 | 30 | 41 | 52 | 63 | 65 | 76 |
| 16 | 27 | 29 | 40 | 51 | 62 | 64 | 75 | 5 |
| 26 | 28 | 39 | 50 | 61 | 72 | 74 | 4 | 15 |
| 36 | 38 | 49 | 60 | 71 | 73 | 3 | 14 | 25 |
| 37 | 48 | 59 | 70 | 81 | 2 | 13 | 24 | 35 |

Magische Zahl = 369

Euklids Algorithmus zur Berechnung des größten gemeinsamen Teilers GGT natürlicher Zahlen:

Seien $x, y \in \mathbb{N}$, $y \neq 0$, $x \geq y$, dann $\exists \alpha \in \mathbb{N}$ mit $x = \alpha y + z$,
damit $\text{GGT}(x, y) = \text{GGT}(y, z)$.

Beispiel:

$$3220 : 79 = 40 \text{ Rest } 60,$$

$$79 : 60 = 1 \text{ Rest } 19,$$

$$60 : 19 = 3 \text{ Rest } 3,$$

$$19 : 3 = 6 \text{ Rest } 1,$$

$$3 : 1 = 3 \text{ Rest } 0.$$

Damit $\text{GGT}(3220, 79) = 1$.

Formulierung des Algorithmus:

Seien $a, b \in \mathbb{N}$,

$x := a$

$y := b$

Solange $y \neq 0$ führe aus

$$z := \left\lfloor \frac{x}{y} \right\rfloor \quad \text{"ganzzahliger Anteil von } \frac{x}{y} \text{"}$$

$$r := x - z * y$$

$$x := y$$

$$y := r$$

$x = \text{GGT}(a, b)$

Bemerkung: (i) Algorithmus terminiert.

(ii) Beweis der Korrektheit einfach.

// Euklids Algorithmus als C++-Programm

```
#include <iostream>
```

```
using namespace std;
```

```
int ggt (int, int); // Größter gemeinsamer Teiler
```

```
int main () {
```

```
    int a = 1; // Erste natürliche Zahl
```

```
    int b = 1; // Zweite natürliche Zahl
```

```
    cerr << "Programm berechnet den größten "
```

```
           "gemeinsamen Teiler zweier natürlicher "
```

```
           "Zahlen"
```

```
           << endl << endl
```

```
           << "Abbruch durch b = 0"
```

```
           << endl << endl;
```

```
    while (b != 0) {
```

```
        cerr << "Erste Zahl a = ";
```

```
        cin >> a;
```

```
        cerr << "Zweite Zahl b = ";
```

```
        cin >> b;
```

```
        int c = ggt (a, b);
```

```
        cout << "GGT ("
```

```
                << a << ", " << b << ") = "
```

```
                << c
```

```
                << endl << endl;
```

```
    }
```

```
    return 0;
```

```
} //main
```

```

int ggt (int x, int y) {
    // Bei fehlerhafter Eingabe wird -2 zurückgegeben.
    if (x < 0)
        return -2;
    if (y <= 0)
        // y = 0 wird hier als fehlerhafte Eingabe betrachtet!
        return -2;
    while (y != 0) {
        int z = x / y;
        int r = x - z * y;
        x = y;
        y = r;
    }
    // Bemerkung: Schleife lässt sich vereinfachen
return x;
} //ggt

```

Beispiel eines Programmlaufs:

Programm berechnet den größten gemeinsamen Teiler zweier natürlicher Zahlen

Abbruch durch b = 0

**Erste Zahl a = 56
Zweite Zahl b = 16
GGT (56, 16) = 8**

**Erste Zahl a = 103
Zweite Zahl b = 19
GGT (103, 19) = 1**

**Erste Zahl a = 19
Zweite Zahl b = 103
GGT (19, 103) = 1**

**Erste Zahl a = 36
Zweite Zahl b = 300
GGT (36, 300) = 12**

**Erste Zahl a = -36
Zweite Zahl b = 300
GGT (-36, 300) = -2**

**Erste Zahl a = 0
Zweite Zahl b = 8
GGT (0, 8) = 8**

**Erste Zahl a = 8
Zweite Zahl b = 0
GGT (8, 0) = -2**

Bemerkung: Die letzten beiden Fälle sollten möglichst das gleiche Resultat liefern. Die Korrektur des C++-Programms ist einfach.

Zusammenfassung:

- **Reale Rechner sind nicht so leistungsfähig, wie man sie sich wünscht.**
- **Es ist einfach, auf einem Rechner beschränkte Verhaltensweisen eines Menschen nachzuahmen.**
- **Das Herzstück der Informatik bilden Algorithmen.**
- **Ein Algorithmus und der Beweis seiner Korrektheit sollten gemeinsam hergeleitet werden.**
- **Die Umsetzung eines Algorithmus in eine Programmiersprache sollte einfach sein.**